AP/2166 8 CN

PTO/SB/21 (04-07)
Approved for use through 09/30/2007. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

TRANSMITTAL FORM

(to be used for all correspondence after initial filing)

Total Number of Pages in This Submission	60

Application Number	10/614,347
Filing Date	July 8, 2003
First Named Inventor	Gregory A. Becker et al.
Art Unit	2166
Examiner Name	Khanh B. Pham
Attorney Docket Number	PA3166US

ENCLOSURES *(Check all that apply)*

- [x] Fee Transmittal Form
  - [x] Fee Attached
- [ ] Amendment/Reply
  - [ ] After Final
  - [ ] Affidavits/declaration(s)
- [ ] Extension of Time Request
- [ ] Express Abandonment Request
- [ ] Information Disclosure Statement
- [ ] Certified Copy of Priority Document(s)
- [ ] Reply to Missing Parts/ Incomplete Application
  - [ ] Reply to Missing Parts under 37 CFR 1.52 or 1.53
- [ ] Drawing(s)
- [ ] Licensing-related Papers
- [ ] Petition
- [ ] Petition to Convert to a Provisional Application
- [ ] Power of Attorney, Revocation Change of Correspondence Address
- [ ] Terminal Disclaimer
- [ ] Request for Refund
- [ ] CD, Number of CD(s) _____
  - [ ] Landscape Table on CD
- [ ] After Allowance Communication to TC
- [ ] Appeal Communication to Board of Appeals and Interferences
- [x] Appeal Communication to TC **(Appeal Notice, Brief, Reply Brief)**
- [ ] Proprietary Information
- [ ] Status Letter
- [x] Other Enclosure(s) (please Identify below):
  1) Confirmation Postcard;
  2) 2 additional copies of Reply Brief;
  3) Check for $250.00.

Remarks

Total page number does not include postcard or check.

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT

Firm Name	Carr & Ferrell LLP Customer Number 22830		
Signature	Kenneth M. Kaslow		
Printed name	Kenneth M. Kaslow		
Date	September 10, 2007	Reg. No.	57,239

CERTIFICATE OF TRANSMISSION/MAILING

I hereby certify that this correspondence is being facsimile transmitted to the USPTO or deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on the date shown below:

Signature	Kenneth M. Kaslow		
Typed or printed name	Kenneth M. Kaslow, Reg. No. 57,239	Date	September 10, 2007

PTO/SB/17 (07-07)
Approved for use through 06/30/2010. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

Effective on 12/08/2004.
Fees pursuant to the Consolidated Appropriations Act, 2005 (H.R. 4818).

FEE TRANSMITTAL
For FY 2007

☑ Applicant claims small entity status. See 37 CFR 1.27

TOTAL AMOUNT OF PAYMENT ($) 250.00

Complete if Known	
Application Number	10/614,347
Filing Date	July 8, 2003
First Named Inventor	Gregory A. Becker
Examiner Name	Khanh B. Pham
Art Unit	2166
Attorney Docket No.	PA3166US

METHOD OF PAYMENT (check all that apply)

☑ Check ☐ Credit Card ☐ Money Order ☐ None ☐ Other (please identify): _____

☑ Deposit Account Deposit Account Number: 06-0600 Deposit Account Name: Carr & Ferrell LLP

For the above-identified deposit account, the Director is hereby authorized to: (check all that apply)

☐ Charge fee(s) indicated below ☐ Charge fee(s) indicated below, **except for the filing fee**

☑ Charge any additional fee(s) or underpayments of fee(s) under 37 CFR 1.16 and 1.17 ☑ Credit any overpayments

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

FEE CALCULATION

1. BASIC FILING, SEARCH, AND EXAMINATION FEES

	FILING FEES		SEARCH FEES		EXAMINATION FEES		
		Small Entity		Small Entity		Small Entity	
Application Type	Fee ($)	Fee ($)	Fee ($)	Fee ($)	Fee ($)	Fee ($)	Fees Paid ($)
Utility	300	150	500	250	200	100	
Design	200	100	100	50	130	65	
Plant	200	100	300	150	160	80	
Reissue	300	150	500	250	600	300	
Provisional	200	100	0	0	0	0	

2. EXCESS CLAIM FEES

Fee Description	Fee ($)	Small Entity Fee ($)
Each claim over 20 (including Reissues)	50	25
Each independent claim over 3 (including Reissues)	200	100
Multiple dependent claims	360	180

Total Claims		Extra Claims	Fee ($)		Fee Paid ($)
_____	- 20 or HP =	_____	x _____	=	_____

HP = highest number of total claims paid for, if greater than 20.

Indep. Claims		Extra Claims	Fee ($)		Fee Paid ($)
_____	- 3 or HP =	_____	x _____	=	_____

HP = highest number of independent claims paid for, if greater than 3.

Multiple Dependent Claims: Fee ($) _____ Fee Paid ($) _____

3. APPLICATION SIZE FEE

If the specification and drawings exceed 100 sheets of paper (excluding electronically filed sequence or computer listings under 37 CFR 1.52(e)), the application size fee due is $250 ($125 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR 1.16(s).

Total Sheets		Extra Sheets		Number of each additional 50 or fraction thereof		Fee ($)		Fee Paid ($)
_____	- 100 =	_____	/ 50 =	_____	(round **up** to a whole number) x	_____	=	_____

4. OTHER FEE(S)

	Fees Paid ($)
Non-English Specification, $130 fee (no small entity discount)	
Other (e.g., late filing surcharge): Reply Brief	250

SUBMITTED BY

Signature	Kenneth M. Kaslow	Registration No. (Attorney/Agent) 32,246	Telephone 650-812-3400
Name (Print/Type)	Kenneth M. Kaslow		Date September 10, 2007

IN THE UNITED STATES
PATENT AND TRADEMARK OFFICE

APPLICANTS:	Gregory A. Becker et al.
SERIAL NO.:	10/614,347
FILING DATE:	July 8, 2003
TITLE:	System and Method for Backing Up a Computer System
EXAMINER:	Khanh B. Pham
ART UNIT:	2166
ATTY. DKT. NO:	PA3166US

CERTIFICATE OF MAILING

I hereby certify that this paper is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Mail Stop Appeal Brief, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on the date printed below:

Date: 9/10/07

Kenneth M. Kaslow

Kenneth M. Kaslow

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

ATTENTION: Board of Patent Appeals and Interferences

REPLY BRIEF (37 C.F.R. § 41.37)

This Reply Brief is being filed within two (2) months of the mailing of the Examiner's Answer mailed on July 10, 2007.

Following is an issue-by-issue reply to the Examiner's Answer.

Sir:

09/14/2007 FHETEKI1 00000060 10614347
01 FC:2402 250.00 OP

This reply brief is submitted in response to the Examiner's answer, mailed on July 10, 2007, rejecting claims 1-12, 14-15, 20, and 26-32 of the above-referenced patent application.

Summary of Claimed Subject Matter (37 C.F.R. § 41.37(c)(1)(v))

With respect to a summary of claim 1, a method for maintaining a backup storage system for a data storage system is provided (See FIG. 3). A plurality of data writes from are received from an application program, the plurality of data writes occurring between a first time and a second time. A backward increment between data on the data storage system at the second time and data on the data storage system at the first time is determined based on the plurality of data writes from the application program to the data storage system. The backward increment is stored. The plurality of data writes is also stored. The backup storage system is then updated so that the data on the data storage system at the second time is the same as the data on the backup storage system at the second time (See, for example, paragraphs [0032] through [0037]).

With respect to a summary of claim 20, a method for using a backup storage system for a data storage system is provided (See FIG. 3). A plurality of data writes captured between an application and the data storage system are received, the plurality of data writes occurring between a first time and a second time. Data blocks in the data storage system that were changed are identified based on the plurality of data writes (See, for example, paragraph [0061] and [0070]). The plurality of data writes are applied to an image on the backup storage system (See, for example, paragraphs [0070], [0087], and [0119]). A forward increment between data on the data storage system at the first time and data on the data storage system at the second time is determined based on the plurality of data writes. A backward increment between data on the data storage system at the second time and data on the data storage system at the first time is also determined based on a plurality of data writes. The forward increment and the backward

increment are stored. The plurality of data writes is also stored. The backup storage system is then updated so that the data on the data storage system at the second time is the same as the data on the backup storage system at the second time (See, for example, paragraphs [0032] through [0037]).

With respect to a summary of claim 26, a system for maintaining a backup storage system for a data storage system is provided. A production intercept layer is configured to receive a plurality of data writes from an application program, the plurality of data writes occurring between a first time and a second time (See, for example, FIG. 2; 240, FIG. 3, 305, and paragraphs [0061] and [0062]). A backup agent is configured to determine a backward increment between data on the data storage system at the second time and data on the data storage system at the first time based on the plurality of data writes from the application program to the data storage system (See, for example, FIG. 3, 354, and paragraphs [0063] through [0067]). A log file container is configured to store the backward increment (See, for example, FIG. 3, 318, and paragraphs [0063] through [0067]). A storage device is configured to store the plurality of data writes (See, for example, paragraphs [0071] and [0072]). A backup manager is configured to update the backup storage system so that the data on the data storage system at the second time is the same as the data on the backup storage system at the second time (See, for example, FIG. 3, 350, and paragraphs [0074] through [0076]).

With respect to claim 20, the Examiner asserts that *Goldstein* teaches a method for using a backup storage system for a data storage system comprising "receiving a plurality of data writes captured between an application and the data storage system, the plurality of data writes occurring between a first time and a second time" at Col. 5 lines 44-48 and Fig. 3; "identifying data blocks in the data storage system that were changed based on the plurality of data writes" at Col 5 lines 23-48; "applying the plurality of data writes to an image on the backup storage system" at Col. 6 lines 6-31; "determining a forward increment between data on the data storage

system at the first time and data on the data storage system at the second time based on the plurality of data writes" at Col. 3 line 55 to Col. 4 line 50 and Figs. 4,6; [and] "determining a backward increment between data on the data storage system at the second time and data on the data storage system at the first time based on a plurality of data writes" at Col. 6 lines 6-31 and Figs. 7-11; "storing the forward increment" at Col. 3 line 55 to col. 4 line 50; "storing the backward increment" at Col. 6 lines 6-31 and FIGS. 7-11; "storing the plurality of data writes" at Col. 6 lines 6-31 and FIGS. 7-11; "and updating the backup storage system so that the data on the data storage system at the second time is the same as the data on the backup storage system at the second time." At Col. 6, lines 6-31 and FIGS. 7-11.

Issue # 1:

The Examiner has rejected Claims 1-12, 14-15, 20, and 26-32 under 35 U.S.C. 102(e) as being unpatentable over Goldstein et al. (U.S. 6,665,815 B1), hereinafter "*Goldstein*."

The Examiner notes that the scope of claim 26 is not the same as claim 20. The Appellants submits that claim 26 was erroneously grouped with group #2 and that claims 26-32 should have been grouped with group #1. The Examiner admits the grouping of claims 26-32 with group #1 in previously issued office actions and the Examiner's Answer (page 8) when the Examiner asserts that "claims 26-32 recite a system for performing a similar method as in claims 1-12, 14-15 and therefore [are] rejected for the same reasons."

Appellants continue to disagree with the Examiner's assertions that *Goldstein* anticipates Appellants' claimed invention and, in the interest of compact prosecution, presents the following arguments.

Group #1: Claims 1-12, 14-15, and 26-32

Examiner's Arguments under 102(e)

With regard to claim 1, the Examiner initially asserted that *Goldstein* teaches a method for maintaining a backup storage system for a data storage system comprising "receiving a plurality of data writes from an application program, the plurality of data writes occurring between a first time and a second time" at col. 5 lines 44-49 and Fig. 3; [and] "determining a backward increment between data on the data storage system the second time and data on the data storage system at the first time based on the plurality of data writes from the application program to the data storage system" at Col. 6 lines 6-60 and Fig. 7; "storing the backward increment" at Col. 6, lines 6-31; "storing the plurality of data writes" at Col. 6, lines 6-31; "and updating the backup storage system so that the data on the data storage system at the second time is the same as the data on the backup storage system at the second time" at Col. 6, lines 6-31.

In the subsequent office action mailed June 30, 2006, the Examiner further asserted that "the difference between the two snapshots is the same as a plurality of data writes applied to the first snapshot because First Snapshot + Data Writes = Second Snapshot." The Examiner went on to assert that "therefore... Second Snapshot – First Snapshot = Data Writes or... the difference between two snapshot = Plurality of Data Writes" and that "Goldstein therefore inherently anticipates the claimed limitation." (*Office action, 7.*)

As Appellants previously argued, the system of *Goldstein* never "receives a plurality of data writes from an application program" as asserted by the Examiner. The *Goldstein* system takes snapshots of a consistent state of data in the data volume disk storage, while system operation is briefly suspended (See col. 3, lines 43-55). By contrast, the

claims set forth in the present application teach intercepting data writes on their way to storage, without suspending operation (See FIG. 3 and specification [0056] through [0089]).

The Examiner makes much of the fact that Appellants admit that the system of *Goldstein* and that of the present invention *sometimes* yield similar results. (Examiner's Answer, 11.) From this argument, the Examiner would presumably also argue that a clock face with no internal workings which always shows the same time would anticipate a working clock, since the two would sometimes agree on the time, i.e., twice a day. It is of course not surprising that the system of *Goldstein* and that of the present invention provide similar results, since both are designed to provide backups from which data may be recovered in the event of a loss. If different backup systems were to result in very different recoveries of data, one would presumably be of little or no use, since it would be providing data much more different from the original data than the other.

But the fact that the systems *may* yield similar results does not support the Examiner's argument, because obtaining different results from the prior art is not a requirement for patentability. It is axiomatic that a new, non-obvious method of accomplishing something is patentable, even if the result is previously known. Indeed, if different results were the test, *Goldstein*, and many of the prior art systems described in *Goldstein*, would never have been allowed.

As previously explained, the *Goldstein* system and the present invention accomplish data backup in very different ways. The two snapshots taught by *Goldstein*, which are virtual copies of the disk volume at two different times, are fundamentally different from a plurality of data writes that are captured from an application program occurring between a first time and a second time, as set forth, in part, in claim 1. Unless new

snapshots are taken after each data write, neither the snapshots nor the difference between the snapshots are sufficient to enable the recovery of data to *any* point in time based on "receiving a plurality of data writes from an application program," as set forth, in part, in claim 1.

While various succedent backups may be made using the difference list and a full base state backup (see *Goldstein* col. 4, lines 26 through col. 5 lines 22), the succedent backups are not accomplished by "determining a backward increment between data on the data storage system at the second time and data on the data storage system at the first time based on the plurality of data writes from the application program to the data storage system" as set forth in claim 1. Rather, *Goldstein* teaches incremental backups based on snapshots and differences between the snapshots, which cannot be equated with the backward increments between data based on the plurality of data writes, as set forth, in part, in claim 1 of the present application.

Further, *Goldstein* fails to teach "storing the backward increment; storing the plurality of data writes; and updating the backup storage system so that the data on the data storage system at the second time is the same as the data on the backup storage system at the second time," as set forth, in part, in claim 1 of the present application. The reverse increment can be archived for later use (see specification [0036]). Instead, *Goldstein* teaches prior art snapshot techniques (see col. 3, lines 44-55) to take snapshots as backups. The difference in data blocks among various snapshots is then utilized to generate succedent and precedent backups. Backward increments are not stored. As discussed herein, *Goldstein* fails to receive a plurality of data writes, which thus cannot be stored in *Goldstein*. In fact, *Goldstein* indicates that the snapshots are typically deleted, once they are rolled out (see FIGS. 5 and 6 and col. 5 line 50 to col. 6 line 5).

The Examiner argues that "the difference between the two snapshots is therefore the same as 'plurality of data writes' applied to the first snapshot." (Examiner's Answer, page 12-13) Appellants disagree. *Goldstein* takes a snapshot and adds additional snapshots. As discussed in the Appeal Brief filed on February 15, 2007, snapshots represent a static capture of the entire contents of a volume at a single point in time, not a "plurality of data writes from an application program to the data storage system."

The Examiner makes much of Appellants analogy of the difference between *Goldstein* and the present invention as being like the difference between snapshots and video, arguing that "taking videos is the same as taking multiple pictures within a short period of time." (Examiner's Answer, 11). But the point of the example was not to say that the present invention functions in such a manner.

Rather, the point of the Appellants' video versus static pictures analogy was to highlight the fact that *Goldstein* cannot recover data to <u>any point in time</u> because *Goldstein* does not "receive a plurality of data writes" nor does it perform or disclose any of the other elements set forth in claim 1. *Goldstein* only captures data at two static times and cannot reproduce what happens to the data in between those two static times. Further, there is no teaching in *Goldstein* that a new snapshot is taken each time there is a data write. Thus, *Goldstein* does not teach all of the series of pictures analogized to "a plurality of data writes" to generate the video discussed in Appellants' analogy, and thus does not yield the result postulated by the Examiner.

Perhaps a better analogy for the difference between *Goldstein* and the present invention is that of recording a chess game. One way to record the game is to take snapshots of the chess board at intervals, a la *Goldstein*. If the pictures are not taken after every move, it is impossible to reconstruct the exact position at any selected time between pictures.

As above, *Goldstein* does not teach taking a snapshot after every move or "data write." However, being overly generous, and accepting the Examiner's argument that "video" is merely a sequence of still pictures, suppose a new picture is taken after every move. Each picture will then reflect the position of the board after a move taken after the preceding picture.

Now consider that instead of taking snapshots, one records the moves by means of a notation that indicates what is being moved and where, a la the data writes of the present invention. For example, in typical notation, the first three moves in the classic "King's Gambit" are:

1. e4 e5
2. f4

It is undeniably true that if one takes a chess board in opening position and follows this notation, one will arrive at the same configurations as shown in the snapshots taken during this play. However, the methods of arriving there are fundamentally different and have different consequences. For example, consider an attempt to store the record of a game so that it may be later reproduced. A hard copy record using notation can fit on significantly less than a sheet of 8-1/2" by 11" paper; putting a game's worth of photographs on such a sheet will probably result in pictures too small to be made out by the naked eye. In electronic media, even using compression techniques to store the snapshots will take significantly more space than storing the notation. Yet the Examiner would apparently argue that they are the same and that one anticipates the other. The absurdity of such an argument is easily apparent.

In fact, *Goldstein* describes its own system as a "physical incremental backup" using "a base state snapshot and a sequential series of data volume snapshots." (*See* at least the title and the abstract). *Goldstein* simply does not capture a plurality of data writes. Instead, *Goldstein* "generates a precedent snapshot difference list, generated by

identifying the data blocks in any snapshot differing from the data blocks in a subsequent snapshot," which is used to recover files without incurring a full restore. The data blocks described by the snapshot difference list are copied to backup storage and the snapshot is deleted and file recovery is accomplished by overwriting data from a current snapshot list with one or more precedent backups. (*See* at least the summary) *Goldstein* describes using a "base snapshot" and a subsequent series of "data volume snapshots", which cannot be equated to "a plurality of data writes from an application program," as set forth, in part, in claim 1. FIG. 3 in *Goldstein* describes snapshots of the data volume of consistent states. Nowhere does *Goldstein* describe "receiving a plurality of data writes from an application program, the plurality of data writes occurring between a first time and a second time." Thus, the Examiner is wrong in asserting that the claimed limitations are anticipated by *Goldstein*.

For at least these reasons, Applicant respectfully submits that *Goldstein* does not anticipate claim 1 as it clearly does not teach or suggest all of the elements of claim 1.

Group #2: claim 20

Claim 20 recites "receiving a plurality of data writes captured between an application and the data storage system, the plurality of data writes occurring between a first time and a second time; identifying data blocks in the data storage system that were changed based on the plurality of data writes; applying the plurality of data writes to an image on the backup storage system; determining a forward increment between data on the data storage system at the first time and data on the data storage system at the second time based on the plurality of data writes; and determining a backward increment between data on the data storage system at the second time and data on the data storage system at the first time based on a plurality of data writes; ." As discussed herein, with

respect to claim 1, *Goldstein* fails to teach "receiving a plurality of data writes captured between an application and the data storage system, the plurality of data writes occurring between a first time and a second time."

As previously discussed, the two snapshots taught by *Goldstein* are virtual copies of the disk volume at those two different times, not a plurality of data writes captured between an application and the data storage system, and occurring between a first time and a second time. *Goldstein* compares the two snapshots to determine a difference list, based on the difference between the two snapshots, as discussed herein.

While *Goldstein* does describe identifying data blocks in the data storage system that changed based on the difference between the two snapshots, *Goldstein* fails to teach "identifying data blocks in the data storage system that were changed based on the plurality of data writes." As in the example of the two pictures of the chair, the data blocks that change between two snapshots is entirely different from the data blocks that change based on the plurality of data writes, or based on a video of the chair actually being moved. As another example, if a data block is written more than once between the time of the first snapshot and the time of the second snapshot, *Goldstein* indicates the last data that was written to the data block, rather than "identifying data blocks in the data storage system that were changed based on the plurality of data writes," as set forth in claim 20.

Goldstein also fails to teach "applying the plurality of data writes to an image on the backup storage system." Specifically, *Goldstein* typically deletes or overwrites the snapshots taken based on new snapshots (see FIGS. 5 and 6). Thus, in addition to the fact that *Goldstein* fails to teach "the plurality of data writes," *Goldstein* appears to apply only current snapshots to any base snapshot it has stored. "To recover the data volume

in the present example, the backups are restored in successive order. The full base state backup 130 (B.sub.0) is obtained and subsequently overwritten with the first succedent backup 131 and then with the second succedent backup 133. This yields an exact copy of the volume as of its second state snapshot 115" (see col. 5, lines 50-54). Further, *Goldstein* teaches concatenating various snapshots. Combining snapshots of a disk volume, even with the difference list, is not the same as "applying the plurality of data writes to an image on the backup storage system," as set forth in claim 20.

Finally, *Goldstein* fails to teach "determining a forward increment between data on the data storage system at the first time and data on the data storage system at the second time based on the plurality of data writes; determining a backward increment between data on the data storage system at the second time and data on the data storage system at the first time based on a plurality of data writes; storing the forward increment; storing the backward increment; storing the plurality of data writes; and updating the backup storage system so that the data on the data storage system at the second time is the same as the data on the backup storage system at the second time." Assuming that "the forward increment" may be determined from the difference list the, *Goldstein* determines a "forward increment" based on the difference between the two snapshots, not based on the plurality of data writes. Assuming that "the backward increment" may be determined from the difference list the, *Goldstein* determines a "backward increment" based on the difference between the two snapshots, not based on the plurality of data writes, as discussed herein. Thus, any increments determined in *Goldstein* do not indicate what happened to the data between the snapshots and thus cannot be utilized to recover the data to any point in time. Further, any increments discussed in *Goldstein* are not stored, but are overwritten, as discussed herein.

Appellants argued that a reverse increment can include individual data writes that can be applied to the application's image--i.e., the production image, replicated image, or read/write snapshot of the replicated or production images, as one transaction to restore the image from the second time back to the first time. The forward increment can then be applied in part, or in its entirety as one transaction to roll the image forward from the first time to any point in time up to and including the second time. Specification [0036]. Thus, while the production image in the present application may include a snapshot, the reverse or forward increments comprising individual data writes, unlike the additional snapshots and difference lists taught by *Goldstein*, are utilized to roll the production image forward or backward to any point in time, rather than to a time determined according to when the snapshot in *Goldstein* is taken.

In the Examiner's Answer mailed on July 10, 2007, the Examiner argues that since *Goldstein* states that an "important characteristic of the physical incremental backup is that only those data blocks that have changed are copied" and that "only the data block containing the updated record, and possibly affected index blocks, are backed up", "Goldstein therefore teaches the step of 'identifying data blocks in the data storage that were changed based on the plurality of data writes.'" However, the disclosure of the copying of data blocks in *Goldstein* does not equate to actually "identifying data blocks in the data storage that were changed based on the plurality of data writes." In fact, as discussed herein, *Goldstein* utilizes snapshots, which are static captures of disk volumes, rather than a plurality of data writes. It is clear error to assert that "data blocks that change between two snapshot is the same as data block that changed based on the plurality of data writes" since the changes between two static disk volumes are different from changes based on captured data writes unless a new snapshot is taken after every data write. In fact, *Goldstein* discloses comparing snapshots to produce a list of blocks that have changed between the snapshots so that those blocks may be copied into backups, the snapshots representing consistent states of the data volume in disk storage (see at least FIG. 1). Clearly, the process disclosed in *Goldstein* is not the same as "receiving a plurality of data writes

captured between an application and the data storage system, the plurality of date writes occurring between a first time and a second time" and "identifying data blocks in the data storage system that were changed based on the plurality of data writes."

Finally, to equate snapshots with data writes is clear error. *Goldstein* admits that a snapshot is generated from a data volume consistent state, rather than capturing data writes "in transit from the application layer... to the storage device layer." (see FIG. 1 of present application)

For at least these reasons, Applicant respectfully submits that *Goldstein* does not anticipate claim 20. The foregoing anticipation criterion has simply not been met by *Goldstein*.

In view of the remarks set forth herinabove, all of the independent claims are deemed allowable, along with any claims depending therefrom.

Respectfully submitted,

Gregory A. Becker et al.

Date: 9/10/07 By: Kenneth M. Kaslow

Kenneth M. Kaslow
Reg. No. 32,246
Carr & Ferrell *LLP*
2200 Geng Road
Palo Alto, CA 94303
TEL: (650) 812-3465
FAX: (650) 812-3444

CLAIMS APPENDIX (37 C.F.R. § 41.37(c)(1)(viii))

Claims on Appeal:

1. A method for maintaining a backup storage system for a data storage system comprising:

receiving a plurality of data writes from an application program, the plurality of data writes occurring between a first time and a second time;

determining a backward increment between data on the data storage system at the second time and data on the data storage system at the first time based on the plurality of data writes from the application program to the data storage system;

storing the backward increment;

storing the plurality of data writes; and

updating the backup storage system so that the data on the data storage system at the second time is the same as the data on the backup storage system at the second time.

2. The method of claim 1, further comprising:

determining a forward increment between the data on the data storage system at the first time and the data on the data storage system at the second time based on the plurality of data writes.

3. The method of claim 2, further comprising:

associating the backward increment with the forward increment.

4. The method of claim 2, further comprising:

storing the forward increment; and

storing an association of the backward increment and the forward increment.

5. The method of claim 1, further comprising:

storing indicia of the plurality of data writes.

6. The method of claim 1, wherein said updating the backup storage system comprises:

applying each of the plurality of data writes to an image of data on the backup storage system, thereby recreating the data on the data storage system at the second time.

7. The method of claim 6, said applying each of the plurality of data writes comprising:

updating the image of the data stored on the backup storage system with the plurality of data writes.

8. The method of claim 1, wherein said updating the backup storage system comprises:

optimally applying the plurality of data writes to the backup storage system, thereby recreating the data on the data storage system at the second time.

9. The method of claim 1, wherein a difference between the first time and the second time is a predetermined time period.

10. The method of claim 1, wherein a difference between the first time and the second time is a variable time period.

11. The method of claim 10, wherein a difference between the first time and the second time is dependent on the rate of the plurality of data writes.

12. The method of claim 7, wherein a difference between the first time and the second time is dependent on a quantity of the plurality of data writes.

13. (cancelled)

14. The method of claim 1, wherein said updating the backup storage system so that the data on the data storage system at the second time is the same as the data on the backup storage system at the second time includes applying the backward increment to an image of data on the backup storage system, thereby recreating the data on the data storage system at the second time.

15. The method of claim 14, wherein said updating the backup storage system so that the data on the data storage system at the second time is the same as the data on the backup storage system at the second time includes applying an individual data write to the image of data on the backup storage system, thereby recreating the data on the data storage system at a point in time between the first time and the second time.

16-19. (cancelled)

20. A method for using a backup storage system for a data storage system comprising:

receiving a plurality of data writes captured between an application and the data storage system, the plurality of data writes occurring between a first time and a second time;

identifying data blocks in the data storage system that were changed based on the plurality of data writes;

applying the plurality of data writes to an image on the backup storage system;

determining a forward increment between data on the data storage system at the first time and data on the data storage system at the second time based on the plurality of data writes;

determining a backward increment between data on the data storage system at the second time and data on the data storage system at the first time based on a plurality of data writes;

storing the forward increment;

storing the backward increment;

storing the plurality of data writes; and

updating the backup storage system so that the data on the data storage system at the second time is the same as the data on the backup storage system at the second time.

21-25. (cancelled)

26. A system for maintaining a backup storage system for a data storage system comprising:

a production intercept layer configured to receive a plurality of data writes from an application program, the plurality of data writes occurring between a first time and a second time;

a backup agent configured to determine a backward increment between data on the data storage system at the second time and data on the data storage system at the first time based on the plurality of data writes from the application program to the data storage system;

a log file container configured to store the backward increment;

a storage device configured to store the plurality of data writes; and

a backup manager configured to update the backup storage system so that the data on the data storage system at the second time is the same as the data on the backup storage system at the second time.

27. The system of claim 26, wherein the backup agent is further configured to determine a forward increment between the data on the data storage system at the first time and the data on the data storage system at the second time based on the plurality of data writes.

28. The system of claim 26, wherein a difference between the first time and the second time is a predetermined time period.

29. The system of claim 26, wherein a difference between the first time and the second time is a variable time period.

30. The system of claim 29, wherein a difference between the first time and the second time is dependent on the rate of the plurality of data writes.

31. The system of claim 26, wherein the backup manager is further configured to apply the backward increment to an image of data on the backup storage system, thereby recreating the data on the data storage system at the second time.

32. The system of claim 26, wherein the backup manager is further configured to apply an individual data write to the image of data on the backup storage system, thereby recreating the data on the data storage system at a point in time between the first time and the second time.